PROGRAMACION CONCURRENTE (Curso 2022/2023)

FECHA ENTREGA TEMPRANA: lunes 24 de abril a las 11:59 pm
FECHA LIMITE ENTREGA: miércoles 3 de mayo a las 11:00 am

Objetivos de la práctica:

1. Programación distribuida y concurrente
2. Socket y ServerSocket en Java

PARTE 1 (no entregable) Implementa un servidor concurrente que espera la conexión de clientes y cuando recibe una solicitud de conexión establece un Socket con el cliente a través del cual recibe el nombre del fichero de texto que dicho cliente quiere obtener, y le envía su contenido a través del flujo de salida asociado al socket. Utiliza un nuevo proceso para llevar a cabo la comunicación descrita tras cada petición nueva de cliente. Modifica la implementación anterior para enviar y recibir a través del canal de comunicación "objetos" de una clase Mensaje que hayas definido previamente. Comprueba que los mensajes se reciben correctamente.

PARTE 2 (entrega) Implementa una aplicación capaz de gestionar y llevar a cabo el intercambio de información entre nodos distribuidos en una red. La **información** a intercambiar podría ser por ejemplo películas, imágenes, ficheros de texto,... elige el tipo de aplicación que quieras cuando implementes la práctica. Utilizaremos en adelante el término genérico información. Se trata de un híbrido entre una arquitectura cliente-servidor y peer-to-peer (p2p) en la que el intercambio de información se realiza directamente entre los propios clientes, y el servidor únicamente actúa proporcionando datos sobre qué información hay disponible en el sistema y quiénes son los clientes propietarios de la información.

Cliente Funcionamiento básico del programa cliente:

- Al iniciar la aplicación se pregunta al usuario por su nombre de usuario.
- Una vez iniciada la sesión, el cliente puede realizar dos tipos de acciones: consultar la información disponible en el sistema, o descargar la información deseada.
- Una vez el usuario elija la información a descargar, comenzará el proceso de descarga (en realidad se descarga directamente de la máquina del usuario propietario) de tal forma que el programa cliente sigue su curso natural, y en particular permitiendo que se realicen otras acciones e incluso otras descargas mientras continua la descarga de la primera información.
- Al margen de la voluntad del usuario, el programa cliente puede actuar como emisor de cualquier información de la que dispone compartida, como propietario de una información que otro cliente solicite. Esta acción será llevada a cabo en un segundo plano permitiendo al usuario continuar con el uso normal de la aplicación.
- Al terminar la aplicación se deberá comunicar el fin de sesión al servidor, y permitir así que éste actualice apropiadamente su base de datos.

Servidor Acciones básicas de la aplicación servidor:

- Al iniciarse, cargará la información de los usuarios registrados en el sistema (si los hay) y todos aquellos datos relativos a éstos que consideres oportunos.
- El servidor atiende de forma concurrente todas las peticiones que realizan los clientes conectados al sistema, en particular:
  - Solicitud de búsqueda de usuarios conectados: El servidor realiza una búsqueda en su base de datos y devuelve los resultados obtenidos.
  - Solicitud de descarga de información: El servidor se comunica con los dos clientes en cuestión, gestionando el inicio de la comunicación p2p entre ellos. Una vez los clientes establecen conexión, el servidor se desentiende de la comunicación p2p.
  - Fin de sesión: Se actualiza apropiadamente la bases de datos.

Guías para la implementación

- *Clase Cliente*: Clase principal de la aplicación cliente que ofrece el soporte para la interacción con el usuario del sistema. Algunos de sus atributos pueden ser información sobre el usuario (ver clase Usuario), los objetos que proporcionan la comunicación con el servidor (socket y flujos).
- *Clase OyenteServidor:* Implementa el interfaz "Runnable" o hereda de la clase "Thread", y será usada para llevar a cabo una escucha continua en el canal de comunicación con el servidor, en un hilo diferente.
- *Clase Usuario*: Guarda información para un usuario registrado en el sistema. Algunos de sus atributos podrían ser el identificador de usuario, dirección ip y lista de información compartida. El servidor almacenará información sobre todos los usuarios registrados en el sistema (instancias de la clase Usuario).
- *Clase Mensaje (abstracta)*: Sirve como raíz de la jerarquía de mensajes que deberemos diseñar. Tiene como atributos al tipo, origen y destino del mensaje en cuestión; e incluye métodos como:

  ```
  public int getTipo();

  public String getOrigen();

  public String getDestino();
  ```

- *Clase Servidor*: Clase principal de la aplicación servidor. Tendrá como atributo una o varias estructuras de datos que contendrán la información centralizada (usuarios conectados, canales de conexión..). El servidor espera la llegada de peticiones de inicio de sesión, y asocia un hilo de ejecución con cada usuario.
- *Clase OyenteCliente*: Implementa el interfaz "Runnable" o hereda de la clase "Thread", y es usada para proporcionar concurrencia respecto a las sesiones de cada usuario con el servidor. El método "run()" se limita a hacer lecturas del flujo de entrada correspondiente, realizar las acciones oportunas, y devolver los resultados en forma de mensajes que serán enviados al usuario o usuarios involucrados.

Será necesario implementar además varias clases adicionales. En particular: clases para cada tipo de mensaje, clases para la interfaz con el usuario, etc. (GUI o de texto).

Para obtener la máxima nota tiene que utilizar TODAS las herramientas que hemos visto en la primera parte del curso para garantizar la corrección en el acceso concurrente a la información compartida por múltiples threads:

- Locks justos (implementados en práctica 2);
- Semáforos (implementados en práctica 3);
- Monitores (implementados en práctica 4);